REMARKS

Claims 1-22 are pending in the application.
Claims 1-22 are rejected.

The office action dated August 12, 2004 indicates that claims 7, 8, 18 and 19 are rejected under 35 USC §112, second paragraph, as being indefinite. These '112 rejections have been rendered moot by the amendments above to claims 7, 8, 18 and 19. In addition, claim 20 has been amended to correct a typo.

The office action also indicates that claims 1-5, 9, 11-16, 20 and 22 are rejected under 35 USC §103 (a) as being unpatentable over Bugnion et al. U.S. Patent No. 6,496,847, and that the remaining claims are rejected under 35 USC §103 (a) as being unpatentable over Bugnion et al. in view of others. These rejections are respectfully traversed.

Certain CPU architectures, such as IA-64, have prohibitions against, and difficulties with, completely saving and restoring the entire context of a guest OS or a host OS. Specifically, certain registers containing context might be corrupted during context switching. If the entire context cannot be saved, the guest OS or host OS might behave incorrectly and crash.

Paragraph 17 of the application provides a simple example. Context of a host OS is contained in registers X, I, J and K. During context switching, however, register X is used to store an address that indicates where the context will be saved. Thus, the register X is overwritten during context switching, whereby the context of register X is lost before it can be saved.

This problem is addressed by the method of claim 1. Claim 1, which has been amended for clarity, recites a method of switching context on a processor, the method comprising saving the context under software control using an inconsequential register; and preventing the processor from changing the context while the context is being saved.

Bugnion et al. don't indicate whether context-switching is done under hardware control or software control. The office action cites col. 4, lines 52-61, but all this passage discloses is that context is saved in a driver.

Bugnion et al. don't teach or suggest preventing the processor from changing the context while the context is being saved. The office action cites passages at col. 11, lines 30-52 and col. 17, lines 18-21. However, these passages simply state that known techniques can be used for context switching (see, specifically, col. 11, lines 40-41; and col. 17, lines 20-21). Bugnion et al. do not elaborate on these known techniques. Moreover, Bugnion et al. state that interrupt entry points can be changed (col. 11, lines 46-49), but do not teach or suggest disabling the interrupts.

Bugnion et al. don't teach or suggest the use of an inconsequential register during context switching. According to paragraph 21 of the application, an inconsequential register is a register that is not used at a predetermined interruption point (PIP).

The office action acknowledges that Bugnion et al. don't identify an inconsequential register, but states that use of an inconsequential register is obvious anyway because the inconsequential register is just another form of memory. However, the office action does not provide reason, motivation or incentive for using an inconsequential register in the manner recited in claim 1. According to MPEP 2143.01 states "Obviousness can only be established by combining or modifying the teachings of the prior art to produce the claimed

invention where there is some teaching, suggestion, or motivation to do so found either explicitly or implicitly in the references themselves or in the knowledge generally available to one of ordinary skill in the art." Bugnion et al. do not identify inconsequential registers, let alone suggesting that inconsequential registers can be used to save context. Moreover, Bugnion et al. appear oblivious to the problem of context corruption during context switching. All Bugnion et al. state is that "Any available memory space may be used to save this information, and actual storage and retrieval may be accomplished using any known technique (see col. 11, lines 39-41). "Any available memory" is vague and offers no reason for identifying registers that are inconsequential and then using the inconsequential registers.

Thus, the office action has not established prima facie obviousness of claim 1. Bugnion et al. don't teach or suggest software-based context switching. Bugnion et al. don't teach or suggest preventing a processor from changing context while the context is being saved. Bugnion et al. don't teach or suggest using an inconsequential register to switch context. Therefore, claim 1 and its dependent claims 2-10 should be allowed over Bugnion et al. alone.

Independent claims 12 and 22 and their dependent claims should be allowed for the reasons above.

On December 1, applicant's attorney Hugh Gortler had a telecon with Examiner Lillian Vo to discuss the office action. The following two questions were posed to the examiner, but were not resolved.

(1) Where do Bugnion et al. provide reason, incentive or motivation for using inconsequential registers?

(2) How do Bugnion et al. prevent a **processor** from changing context?

Regarding point (1), the examiner maintains that an inconsequential register is just another form of memory that can be used to store context. However, the statement is inaccurate. The statement is inaccurate because the specification states that the inconsequential is not just any memory. At paragraph 21, the specification describes an inconsequential register as a register that does not store context at a predetermined interruption point (PIP) This feature is elaborated upon in new claim 25.

The statement is also inaccurate because it is inconsistent with the language of claim 1. Claim 1 doesn't recite storing context in an inconsequential register, but rather using the inconsequential register to store context (this feature is clarified in claim 26). The consequential register can be used to store context, for example, by storing an address that indicates where the context will be saved (this example is recited in new claim 24).

It was pointed out during the telecon that the contents of the inconsequential register can be corrupted during context switching (this feature is recited in new claim 23). If the inconsequential register can be corrupted during context switching, it makes no sense to store context in it.

It was also pointed out during the telecon that Bugnion et al. provide no reason or motivation for using an inconsequential register as opposed to other memory. The lack of reason/motivation/incentive is discussed above.

It was also pointed out during the telecon that the application discusses a problem that results from context being corrupted during context switching. The teachings of Bugnion et al. (using any memory) do not overcome the problem.

Regarding point (2), the examiner did not cite any specific passages in Bugnion et al. where a processor was prevented from switching context. However, the examiner appears to have stated that the process of switching context is what prevents the context from being changed.

This statement makes no sense. First, context switching is performed to save old context and replace it with new context. Thus, context in the register is

changed (the old context can be subsequently restored). If the examiner stands by this statement, she is respectfully requested to find support in the prior art.

The rejection of claim 11 was not discussed during the telecon. However, claim 11 has been amended for clarity. Amended claim 11 recites a method of switching context between a host OS and a virtual machine on a processor. The processor has privileged registers and it has access to other memory. The method comprises giving the virtual machine access to the privileged registers; using at least one privileged register to save the context in the other memory at a predetermined interruption point; and preventing the processor from changing the context while the context is being saved. The virtual machine application controls the context switch. Claim 11 should be allowed for several of the reasons above.

The examiner is respectfully requested to withdraw the rejections of the claims, and issue a notice of allowability. The examiner is invited to contact the undersigned to discuss any remaining issues.